Swarthmore College

Works

2023

The Explorer–Director Game On Graphs

Pat Devlin
Swarthmore College, pdevlin2@swarthmore.edu

E. Meger

A. Raz

See next page for additional authors

Recommended Citation

Authors

Pat Devlin, E. Meger, A. Raz, and Polymath REU Participants

The Explorer-Director Game on Graphs

Pat Devlin,[*] Erin Meger,[†] Abigail Raz,[‡] Polymath REU Participants[§]

April 20, 2021

Abstract

The Explorer-Director game, first introduced by Nedev and Muthukrishnan, can be described as a game where two players—Explorer and Director—determine the movement of a token on the vertices of a graph. At each time step, the Explorer specifies a distance that the token must move hoping to maximize the amount of vertices ultimately visited, and the Director adversarially chooses where to move token in an effort to minimize this number. Given a graph and a starting vertex, the number of vertices that are visited under optimal play is denoted by $f_d(G, v)$.

In this paper, we first reduce the study of $f_d(G, v)$ to the determination of the minimal sets of vertices that are *closed* in a certain combinatorial sense, thus providing a structural understanding of each player's optimal strategies. As an application, we address the problem on lattices and trees. In the case of trees, we also provide a complete solution even in the more restrictive setting where the strategy used by the Explorer is not allowed to depend on their opponent's responses. In addition to this paper, a supplementary companion note will be posted to arXiv providing additional results about the game in a variety of specific graph families.

1 Introduction

Consider a game with two players—an Explorer and a Director—played on a graph G. The game is played by moving a token between the vertices of G in the following manner: on any given turn of the game, the Explorer selects a distance to move the token, and the Director goes on to move the token to a vertex that is that distance away from the token's present position. The goal of the Explorer is to maximize the number of vertices the token visits, while the Director's goal is to minimize that same number. The game generally ends when

---

[*]Department of Mathematics, Yale University, patrick.devlin@yale.edu

[†]Laboratoire d'algèbre, du combinatoire et d'informatique mathématique, Université du Québec à Montréal, mckenna_meger.erin_kathleen@courrier.uqam.ca

[‡]Department of Mathematics, University of Nebraska, Lincoln, araz2@unl.edu

[§]Ben Bates, Tyler Beauregard, Daniel Brey, Elaine Danielson, Mayur Lahane, Angela Li, Mohamed Lotfi, Stephanie Marsh, Oskar Mathiasen, Audrey McMillion, Dhruva Mehta, Joseph Mescall, Jonah Nan, Jarod Palubiski, Cac Phan, Isabel Sacksteder, Danielle Solomon, João Tavares, Lincoln Updike, Shannon Vogel, James Wash, Charles Weng, Erica Zhang, Mengxue Zhang

the Explorer cannot effectively call any sequence of distances that would force a visit to an unvisited vertex.

First introduced by Nedev and Muthukrishnan as the Magnus-Derek Game, this game is intended to simulate the behavior of a *Mobile Agent (MA)* exploring a ring network where there is inconsistent global sense of direction [8]. Through focusing on the specific case in which the game is played on a circular table with n positions labelled consecutively, Nedev and Muthukrishnan had established that the number of vertices of this cycle visited assuming optimal play, which they denoted $f^*(n)$, would be:

$$f^*(n) = \begin{cases} n & \text{if } n = 2^k \text{ for some } k > 0 \\ \frac{n(p-1)}{p} & \text{if } n = pm, \text{ where } p \text{ is the smallest odd prime factor of } n \text{ and } p > 2 \end{cases}$$

Follow-up research on the game, such as the ones done by Hurkens et al., Nedev, and Chen et al., has advanced alternative algorithms that achieve $f^*(n)$ more quickly with reduced computational complexity [3, 6, 7, 1]. Chen et al. have also developed strategies for other game variants, while others like Gerbner brought about a new angle by introducing an algebraic generalization for the original game, in which positions $g \in G$ are seen as elements of some cyclic group, G [1, 2]. Asgeirsson and Devlin further built on this idea [9]. Through relating the game to combinatorial group theory and the notion of *twisted subgroup*, they expanded the scope of the game beyond cyclic groups to include other general finite groups, and named the game as the Explorer-Director Game – the generalized version of the original game which we will be focusing on for our research [9].

In this paper, we make progress by developing strategies in various types of graphs, introducing new variants of the game, and constructing alternative algorithms which determine game completion and the minimal solution.

In a graph G, the distance between a pair of vertices $u, v \in V(G)$ to be the length of the shortest path between them. For the explorer director game on a graph G with starting vertex v, we define the explorer-director number, $f_d(G, v)$, as the number of visited vertices at the end of the game.

We begin in Section 2, where we introduce the notion of a *closed set* which in turn implies generic bounds on $f_d(G, v)$. We follow this with Section 3 where we provide specific bounds for lattices and trees. Our final results end with Section 4 where we consider a nonadaptive strategy. In a subsequent arXiv companion, we include additional results for specific graphs including a full solution for triangular lattices. For ease of reading, these results have been separated from the main results presented here. The results in the arXiv companion allow insight into the game by highlighting important strategies used. We have included the most useful and novel techniques and game strategies in this paper.

An interesting result that shows why we care to look at classes of graphs. These values are not bounded based on subgraphs or vertex numbers; so different classes of graphs could give bounds on larger classes.

Corollary 1.1. *For any $b \geq 2$ and $2 \leq k \leq b$, there exists a connected graph G on b vertices such that $f_d(G) = k$.*

For the simple existence of this, consider a lollipop graph with an $b - k + 2$ complete graph and an $k - 2$-path graph. There are a number of constructions that give this result.

2   Closed Subsets

In this section, we investigate the structure of the set of visited vertices at the end of an optimally played game.

Definition 2.1. *On a graph $G = (V, E)$, we define a non-empty subset $U \subseteq V$ to be* closed *when every vertex $u \in U$ has the property that for every vertex $v \in V$, there exists some vertex $x \in U$ such that $dist(u, v) = dist(u, x)$.*

Our notion of closed sets is very similar to another notion of "distance closed" sets (see [4]), where the distances are calculated only using the induced subgraph on U. We allow all distances to be calculated using the ambient graph.

In the Explorer Director game, whenever the token is on a vertex of a closed set, regardless of the distance called by the Explorer, the Director can always find a vertex in the closed set to visit. Notice that closed sets exist in every graph, by using the trivial closed set $U = V$, and by definition the empty set is not closed.

Theorem 2.2. *For any graph G we have $\min_{v \in V} f_d(G, v)$ is the cardinality of the minimal closed set in G.*

As the Director can always keep the token within a closed set we know that $f_d(G, v)$ is at most the cardinality of the minimal closed set containing v. The lower bound is implied by the following more general result.

Theorem 2.3. *At the end of the Explorer-Director game, the set of all visited vertices, U, always contains a closed subset.*

Proof. Let U be the set of visited vertices at the end of the Explorer-Director game on a graph $G(V, E)$. Let X be the set of all vertices $u \in U$ such that there is some vertex $v \in V$ such that for every vertex $w \in U$ we have $\text{dist}(u, v) \neq \text{dist}(u, w)$. Thus $X = \emptyset$ if and only if U is closed. Setting $U = U_1$ and $X = X_1$ let $U_{i+1} = U_i \setminus X_i$ for all $i \geq 1$ (where X_i is the set of all vertices $u \in U_i$ such that there is some vertex $v \in V$ such that for every vertex $w \in U_i$ we have $\text{dist}(u, v) \neq \text{dist}(u, w)$).

We claim that if $v \in \bigcup_{i \leq n} X_i$ and the token is on vertex v then the Explorer can force the token out of U in at most n steps. Clearly, this claim holds for $n = 1$ by the definition of X. We will assume, for the sake of induction, that the statement holds for all vertices in $\bigcup_{i \leq k} X_i$. Assume the token is on some vertex $v \in X_{k+1}$. Note that by definition of X_{k+1} the Explorer can force the token out of U_{k+1} in one move. Additionally, as $U \setminus U_{k+1} = \bigcup_{i \leq k} X_i$ our inductive hypothesis implies that the Explorer can force the token out of U in at most $n + 1$ steps, as desired.

Now we claim that U_n is closed for some n, which implies our result as $U_n \subseteq U$ for all n. If this is not the case then $U_n = \emptyset$ for some n. However, this then implies that $U = \bigcup_{i \leq n} X_i$, so by our previous paragraph if the token is in any vertex in U then the Explorer can force the token out of U in at most $n + 1$ steps. This contradicts our assumption that U is the set of visited vertices at the end of a game. Thus there must be some $U_n \subseteq U$ which is closed, as desired. □

Theorem 2.3 has the following nice corollary which provides a lower bound on $f_d(G)$.

Corollary 2.4. *If in a game there exists a set of vertices $A \subseteq V$ which the Explorer can force the Director to visit at least once, then $f_d(G) \geq \min_{v \in A} ecc(v) + 1$.*

This simply follows from Theorem 2.3 and the fact that for any vertex v any minimal closed set containing v must have size at least $ecc(v) + 1$.

3 Specific Graphs

The problem of the Explorer-Director game was originally defined on an additive group modulo n, which is isomorphic to playing the game on a cycle with n vertices [8]. Here we further explore additional graph families. Additional results on other graph families can be found in the arXiv companion.

3.1 Lattices

Let $L_{n,m}$ denote an n by m rectangular lattice with n vertices in each row and m vertices in each column, so that $|V(L_{n,m})| = nm$. Since $L_{n,m}$ is isomorphic to $L_{m,n}$ we will always assume $n \geq m$. We will specify a vertex v by (x, y) to indicate its position in the lattice when arranged in the usual way in the plane. That is, the bottom-left corner will be $(1, 1)$, the top-left corner will be $(n, 1)$, and the top-right corner will be (n, m).

Theorem 3.1. *For the graph $L_{n,m}$ with $n, m \geq 2$ if one of n or m is odd, then for any starting vertex v,*

$$f_d(L_{n,m}, v) = n + m - 1$$

Otherwise, we have

$$n + m \leq f_d(L_{n,m}, v) \leq 2m + n - 2$$

In a few small cases when n and m are both even we know the lower bound holds with equality. For instance, $L_{2,2} \cong C_4$ and so we know $f_d(C_4, v) = 4$. It is also simple to show that $f_d(L_{4,4}, v) = 8$ if v is along one of the outer sides of the lattice. The Explorer can force the token to at least 8 distinct vertices and that the top and bottom rows form a closed set as do the left and right-most columns. However, we do not know the exact answer in general.

Proof. Regardless of the size of the lattice and starting vertex Corollary 2.4 provides the lower bound since given any starting vertex the explorer can call the eccentricity of that vertex to force the token to a corner and $n + m - 2$ is the eccentricity of any corner vertex. When playing on $L_{n,m}$, all vertices that will be visited can be predetermined by the Director regardless of the Explorer's choices. Director will have two strategies based on the parity of n and m.

We will first focus on the case when one of n and m are odd. Note that in this case there are at most 2 vertices of minimum eccentricity; we will call these central vertices. Given any starting vertex v let a be a corner vertex furthest from v and let b be the corner vertex

antipodal to a. Let A be a path of length $n + m - 1$ which passes through v and the central vertices and has endpoints a and b. See Figure 1 for two examples of such a path (the central vertex or vertices are labeled with c).
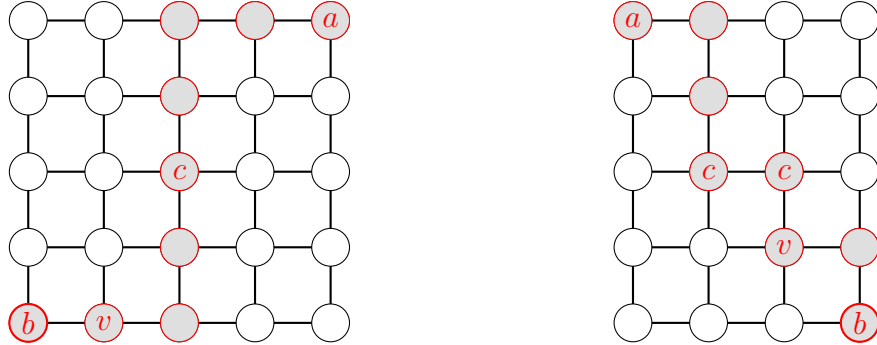

Figure 1: An example of a path A on the lattices $L_{5,5}$ and $L_{5,4}$

It is not too hard to check that such a path will always be a closed set proving that $f_d(G, v) \leq n + m - 1$.

We now consider the case when both n and m are even. The key difference is that now $L_{n,m}$ has four central vertices making up a central square. This change allows the Explorer to force the token to visit all four corners, while in the previous case the Director could always ensure only two corners are visited.

Here we can slightly improve the $n + m - 1$ lower bound implied by Corollary 2.4. Given a particular vertex v let $N_d(v)$ be all the vertices of distance d from v. Let u and w be two non-antipodal corners in $L_{n,m}$. Note that the sets $N_0(u), N_2(u), \ldots, N_{n+m}(u), N_0(w), N_2(w), \ldots,$ $N_{n+m}(w)$ are all disjoint (we are only looking at vertices of even distance from either u or w). Since the Explorer can ensure all corner vertices are visited they can also ensure at least one vertex from each of these $n + m$ disjoint sets are visited implying $f_d(L_{n,m}, v) \geq n + m$ for any starting vertex v.

In this case our upper bound comes from noting that for any given starting vertex $v = (x, y)$ the set

$$A = \{(i, 1) : 1 \leq i \leq m\} \cup \{(i, n) : 1 \leq i \leq m\} \cup \{(x, i) : 1 \leq i \leq m\}$$

is closed. See Figure 2 for an example of such a set. Note that this set clearly has size $2m + n - 2$ and contains v giving $f_d(L_{n,m}, v) \leq 2m + n - 2$.

We can do slightly better when $n = m$ and the starting vertex is in a particular position. In this case, for instance, the set

$$B = \{(x, 1) : 0 \leq x \leq n\} \cup \{(x, n) : 0 \leq x \leq n\} \cup \{(n - 1, x) : 3 \leq x \leq n - 2\}$$

is closed and of size $3n - 4$. Thus if $v \in B$ we have $f_d(L_{n,n}, v) \leq 3n - 4$. We show an example of the set B for the lattice $L_{6,6}$ in Figure 3.
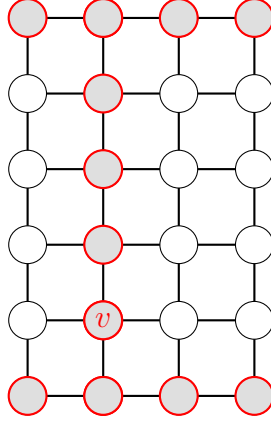
□

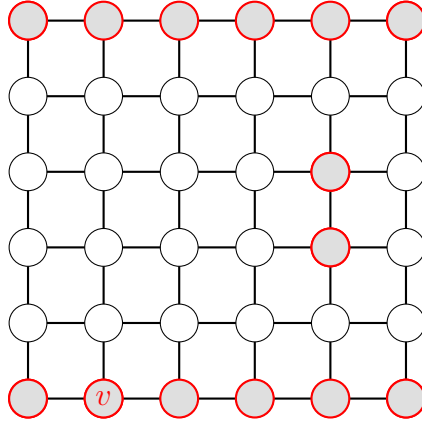Figure 2: An example of the set A on the lattice $L_{6,4}$

Figure 3: An example of the set B on the lattice $L_{6,6}$

3.2 Trees

In this section, we consider the Explorer-Director game played on a tree. We will give an equality bound for $f_d(G, v)$, for any starting vertex $v \in V(G)$ in any tree G.

First, we consider the following lemma to provide a lower bound and introduce the strategy we will use in the proof of the main result. In optimal play, the Director will always return to a visited vertex whenever possible. In the following strategy, we show how the Explorer can take advantage of this optimal strategy for the Director.

Lemma 3.2. *Let G be a tree. Then for all vertices v of G, $f_d(G, v) \geq diam(G) + 1$.*

Proof. Let the diameter of the tree be d. Consider a set of vertices A such that for each vertex in $a \in A$ there exists some vertex v such that $\text{dist}(a, v) = i$ for each i in $\{1, ..., d\}$. The set A must contain at least two vertices: the two endpoints of a path along the diameter. Every vertex in A must be a leaf.

Whenever the Director occupies a vertex in A, there exist at least d vertices with distinct distances, thus the Explorer can force the Director to visit at least one vertex of each distance.

6

Whenever the Director occupies a vertex outside of A, the Explorer can force the Director back into A, and the game-play continues in this manner until $d + 1$ vertices have been visited. There are $d + 1$ vertices along the diameter path.

□

Throughout this section, we will use the strategy of moving from a set of vertices to a new vertex, and then returning to the initial set. In this way, the Explorer can force the Director into new vertices. We say that $\mathcal{P}$ is the set of all paths P in G that have length $\text{diam}(G)$, and use this notation in the rest of the section.

Remark 3.3. *Above, we have shown that the Explorer can force the director to visit all the vertices on a specified diameter path. In an optimal strategy, the Director will always remain on a particular path $P \in \mathcal{P}$. Otherwise, the set of visited vertices will included the final diameter path that the Explorer has forced, as well as the initial vertices the Director had visited.*

Considering Remark 3.3, we now consider the main result of the section. When playing the original geodesic distance version of the Explorer-Director game, we have found the precisely the number of vertices that must be visited in optimal gameplay.

Theorem 3.4. *Given a tree G with set of diameter paths $\mathcal{P}$. For a starting vertex $v \in V(T)$, let $u \in V(P)$ for some $P \in \mathcal{P}$, and we let $\ell = \min_{u \in P \in \mathcal{P}} \text{dist}(u, v)$. Then,*

$$f_d(G, v) = \text{diam}(G) + \ell + 1$$

Proof. We begin with the proof of the upper bound. Let G be a tree with set of diameter paths $\mathcal{P}$, and specified started vertex v.

First we note, that whenever the Director occupies a vertex, v on some diameter P of the tree G, they will always remain on vertices in P, since there exists a vertex on P of every distance up to $\text{ecc}(v)$, thus remaining on P will minimize the number of vertices visited. Whenever the Director occupies a vertex that is not on P for some diameter path $P \in \mathcal{P}$, they will choose to move towards some such P. Once they have entered some path P, they will never leave, and thus the most number of vertices that can be visited are the number of vertices in P, which is $\text{diam}(G) + 1$, along with the initial vertices required to reach P, which is at most ℓ as defined in the statement of the theorem, thus giving the upper bound.

Consider now the optimal strategy of the Explorer. Let $u \in V(P), P \in \mathcal{P}$ be the particular vertex closest to v that is on a diameter path. For each vertex along the geodesic between u, v, we claim the explorer can force a visit to each vertex. Beginning on v, the Explorer uses a similar strategy as used in Lemma 3.2. For each $i \in \{1, ..., \text{dist}(u, v)\}$, the Director will move towards u to minimize the number of vertices visited. The Explorer however cannot continue to call 1 each turn, since the Director would then move back to the previously visited vertex v. The Director will never move to a vertex off this path in order to minimize the number of vertices visited. If the Director chooses a vertex not on the geodesic between u and v, the Explorer can continue this strategy, and force the Director to visit even more vertices. Thus, the Explorer must return to v after each new vertex along the path of $\text{dist}(u, v)$, and can force the Director to visit one new vertex every other round. This completes the lower bound.

□

4   Nonadaptative strategies

In this section, we consider a case in which the Explorer has a strategy which is independent of the Director's choices. We require the following preliminary definition.

Definition 4.1. *We define a nonadaptative strategy S as a finite sequence of natural numbers, and its score denoted $f_d^S(G, v)$ as the minimum number of visited vertices if the explorer plays according to S starting at v. We say a nonadaptive strategy, S, is optimal if $f_d^S(G, v) = f_d(G, v)$.*

There does not necessarily exist an optimal nonadaptative strategy for trees when the starting vertex is not in a longest path. For example, consider the following tree, where C is the center of a sufficiently long path.
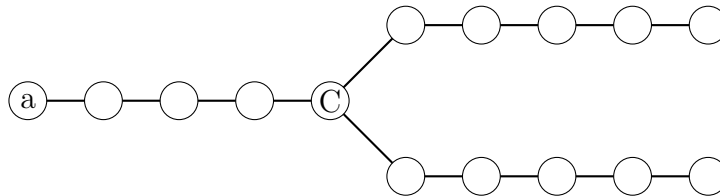
Note that by Theorem 3.4 we know $f_d(G, a) = |V(G)|$ for such a graph. However, in any nonadaptive strategy if the explorer ever plays a distance other than 1, then the director can ensure that some vertex in G is never visited. Clearly, if the director only ever plays distance 1 then the explorer can restrict the token to only a and its neighbor. Thus there is no optimal nonadaptive strategy for this graph.

We will show below that for a particular class of trees there is an optimal nonadaptive strategy when we start at a vertex in a longest path.

Recall that $N_k(v)$ denotes all vertices of distance k from v and any vertex of minimal eccentricity is a central or center vertex. We will use the following standard proposition on trees see, for example, [5].

Proposition 4.2. *A tree has at most two centers.*

For the following results, we focus on trees with only one center, however, the case for bi-centered trees is analagous.

Theorem 4.3. *Let T be a tree with center c. If every component in $T \setminus \{c\}$ contains at most one endpoint of a longest path of T and the starting vertex, v is contained in some longest path then there is an optimal nonadaptive strategy.*

Proof. Let us assume, without loss of generality, that the starting vertex is an endpoint, otherwise the explorer starts by calling the eccentricity of the starting vertex. Let $L > 3$ be the length of the longest path and R the radius, that is the eccentricity of the center. The $L \leq 3$ case is simple to check. Note that by Theorem 3.4 $f_d(G, v) = L + 1 = 2R + 1$ for v being an end point.

We define the subsequence $s_i = (i, L - i, i, L - i)$ and we define our strategy as the concatenated subsequences $S = (s_1, s_2, \ldots, s_{R-1}, R)$. We claim $f_d^S(G, v) = L + 1$.

We will show that after s_1 is called 4 vertices will have been explored, after each subsequent s_i is called two new vertices are explored, and the token will visit the center on the last move.

If the token is at an endpoint of a longest path then for any $1 \le i \le R-1$ we know that after the Explorer calls i followed by $L-i$ the token will again be at a different endpoint of a longest path. Thus after s_1 is called we know the token must have visited 4 distinct vertices. Furthermore, we know that after calling s_i the token will have visited at least two vertices that were not visited in the first s_{i-1} subsequences. This is because after i is called the first time in the s_i subsequence the token must move to a vertex of distance i from an endpoint and after $L-i$ is called the token must return to an endpoint of a longest path. Therefore, before the last move we have explored $4 + 2(R-2) = 2R$ vertices and the token is at an endpoint. Finally, the explorer calls R the token is forced to the center, and the number of visited vertices is now $2R + 1 = f_d(G, v)$ as desired. $\qquad\square$

The above shows that for a variety of trees, the Explorer can use a completely nonadaptive strategy to visit the optimal number of vertices. However, the strategy provided requires the Explorer to revisit the same leaves many times, so essentially only half the steps take us to new vertices. In the case of paths, however, we show that these revisits are unnecessary.

Theorem 4.4. *If G is a path, then the Explorer has a nonadaptive strategy that visits each vertex exactly once, regardless of the Director's choices.*

Proof. We provide a construction showing that for any starting vertex, the Explorer can efficiently explore the path by repeatedly declaring numbers that are so large that the Director never has any choice for where the token should go. We show two cases for the parity of the length of the path. If the path has an odd number of vertices, then the Director would necessarily have a choice whenever the token is on the center vertex.

We first address the case that G has an even number of vertices, labeled $v_1, v_2, \ldots, v_{2k}$. The strategy is to declare distance 1 when the token is at v_1, and afterwards to alternate between declaring distances k and $k-1$, as shown in Figure 4.
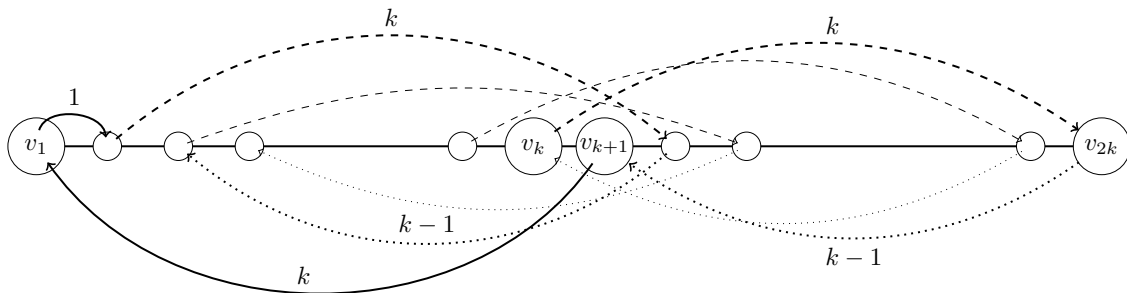

Figure 4: Nonadaptive strategy in even paths, alternating steps of size k (dashed, drawn above the path) and $k-1$ (dotted, drawn below)

Each time the distances k or $k-1$ are declared, the Director always has exactly one choice for where to move the token, since the distance k is declared at each of the two center

9

vertices. Moreover, as can be seen, the parity of the path conveniently makes it so that if the token starts at v_1, then by the time it arrives at v_k, the only unvisited vertices will be v_{2k} and v_{k+1}. Thus, declaring distance k followed by $k-1$ forces the token to each of these in turn.

When the path has an odd number of vertices, the unique center vertex prohibits us from being able to simply force the token as before, and our strategy is necessarily more involved. For this, we number the vertices $v_0, v_1, \ldots, v_{2k}$, with v_k being the center vertex. We begin at a vertex v_x with $0 \le x < k$ and we exhibit a forcing strategy that visits each vertex exactly once and ending on v_k. If—on the other hand—the vertex instead starts at v_k, notice that the Explorer could start by saying any distance they like and then simply follow our strategy, omitting the final step of moving to v_k. Our strategy is described in two phases, as shown in Figure 5.
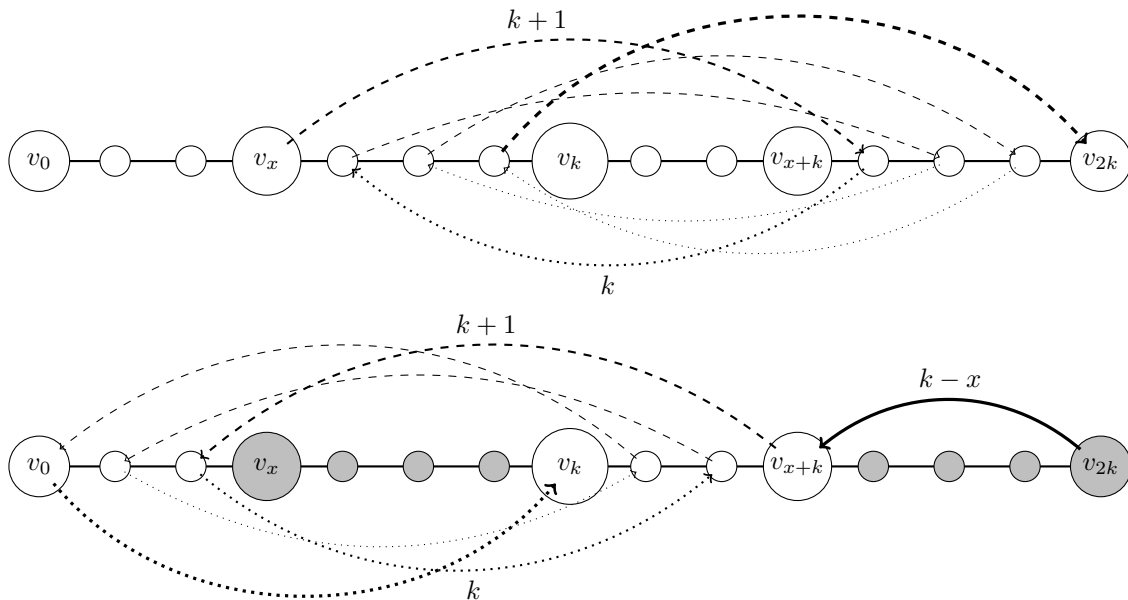

Figure 5: Nonadaptive strategy for odd paths, shown in two phases. Essentially alternating steps of size $k+1$ (dashed, drawn above the path) and k (dotted, drawn below)

In the first phase, the token begins at v_x, and the Explorer alternates between declaring $k+1$ and k until the token arrives at v_{2k}. At this point, the token will have seen every vertex v_t where $x \le t < k$ as well as vertices where $x + k < t \le 2k$, and the token is at v_{2k} (as shaded gray in Figure 5).

The second phase of the strategy begins with the token at v_{2k}, and the Explorer declares $k-x$ to move the token to v_{x+k}. After this, the Explorer simply alternates between declaring $k+1$ and k (mirroring the path in phase one) until they arrive at v_0. Finally, declaring k moves the token to v_k, which is the last vertex that needed to be visited. $\qquad \square$

5 Conclusion and Further Research

We conclude this paper with a discussion on further research. We next define a variant using distance norms for the director to move, such as allowing the director to move along any path, denoted $f_p(G, v)$ or even trail, denoted $f_t(G, v)$ a given length. Notice that a walk would not be an interesting variant. Interestingly, the number of visited vertices when allowing the token to move along paths, $f_p(G, v)$ can be higher than $f_d(G, v)$ as the Explorer can now call significantly higher values.

Another variant we propose is that of a weighted graph. It is shown in the arXiv Companion that every graph has a weighting such that the set of vertices visited at the end of the game is the entire graph. Further work in this variant could include determining the complexity, adding constraints on the vertex weights, or classifying the sets of weights that allow all vertices to be visited in terms of other graph parameters.

One can also consider the number of steps required to visit $f_d(G, v)$ vertices for various graph families, as was done in Theorem 4.4 as well as [3, 6, 1, 7]. This then furthers the research into determining the complexity of the problem. Similarly, further work into what graph families allow for non-adaptive strategies is of interest. Finally, there are clearly many more interesting graph families to consider such as hypercubes or even random graphs.

Acknowledgements

References

[1] L.-J. Chen, J.-J. Lin, M.-Z. Shieh, and S.-C. Tsai. More on the magnus-derek game. *Theoretical Computer Science*, 412(4):339 – 344, 2011.

[2] D. Gerbner. The magnus-derek game in groups. *Discret. Math. Theor. Comput. Sci.*, 15:119–126, 2013.

[3] C. A. Hurkens, R. A. Pendavingh, and G. J. Woeginger. The magnus-derek game revisited. *Information Processing Letters*, 109(1):38 – 40, 2008.

[4] T. Janakiraman, P. Alphonse, and V. Sangeetha. Structural properties of k-distance closed domination critical graphs for k= 5 and 6. *International Journal of Engineering Science Advanced Computing and Bio-Technology*, 3:1–14, 2012.

[5] D. E. Knuth. *The Art of Computer Programming, Volume 1 (3rd Ed.): Fundamental Algorithms*. Addison Wesley Longman Publishing Co., Inc., USA, 1997.

[6] Z. Nedev. An o(n)-round strategy for the magnus-derek game. *Algorithms*, 3:244–254, 09 2010.

[7] Z. Nedev. A reduced computational complexity strategy for the magnus-derek game. *International Mathematical Forum*, 9:325–333, 2014.

[8] Z. Nedev and S. M. Muthukrishnan. The magnus-derek game. *Theoretical Computer Science*, 393(1):124 – 132, 2008.

[9] D. T. Ásgeirsson and P. Devlin. Palindromes in finite groups and the explorer-director game, 2019.